Praktické aspekty vývoje softwaru - zpráva o profilování

1 Úvod

V projektu jsme měli využít naši matematickou knihovnu, a pomocí jejích funkcí vypočítat výběrovou směrodatnou odchylku pro deset, sto a tisíc čísel.

Pro profilování jsme využili utilitu `cProfile`, který je vestavěný ve standardní instalaci Python. Pro vizualizaci jsme použili IDE `PyCharm`.

2 Výsledky

Při malém množství vstupních hodnot vidíme, že většinu času spotřebovalo samotné spuštění skriptu. Vidíme že na prvních 25 pozicích není ani jedna matematická operace.

Name	Call Count	Time (ms)	Own Time (ms)
smerodatna_odchylka.py	1	839 100,0%	0 0,0%
_find_and_load_unlocked	6	793 94,5%	0 0,0%
_find_and_load	6	793 94,5%	0 0,0%
_load_unlocked	6	790 94,2%	0 0,0%
math.py	1	686 81,8%	0 0,0%
decimal.py	1	617 73,5%	0 0,0%
module_from_spec	6	615 73,3%	0 0,0%
<built-in method io.open>	2	157 18,7%	157 18,7%
_find_spec	6	40 4,8%	0 0,0%
find_spec	6	40 4,8%	0 0,0%
_get_spec	6	39 4,6%	0 0,0%
find_spec	29	38 4,5%	0 0,0%
_path_stat	46	37 4,4%	0 0,0%
<built-in method nt.stat>	46	37 4,4%	37 4,4%
_path_is_mode_type	12	31 3,7%	0 0,0%
_path_isfile	10	31 3,7%	0 0,0%
square	1	13 1,5%	13 1,5%
_new_module	5	0 0,0%	0 0,0%
__init__	6	0 0,0%	0 0,0%
acquire	6	0 0,0%	0 0,0%
release	6	0 0,0%	0 0,0%
__init__	6	0 0,0%	0 0,0%
__enter__	6	0 0,0%	0 0,0%

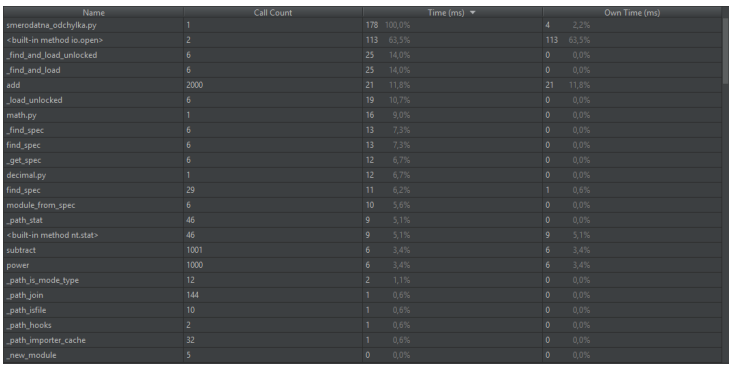
Obrázek 1: Výstup z profilování s 10 vstupními hodnotami

Při spuštění s více vstupními hodnotami vidíme, že se dopředu posouvají matematické operace, které nejčastěji používáme.

Name	Call Count	Time (ms)	Own Time (ms)
smerodatna_odchylka.py	1	49 100,0%	0 0,0%
<built-in method io.open>	2	26 53,1%	26 53,1%
_find_and_load_unlocked	6	20 40,8%	0 0,0%
_find_and_load	6	20 40,8%	0 0,0%
_load_unlocked	6	15 30,6%	0 0,0%
math.py	1	13 26,5%	0 0,0%
_find_spec	6	10 20,4%	0 0,0%
_get_spec	6	10 20,4%	0 0,0%
find_spec	6	10 20,4%	0 0,0%
decimal.py	1	10 20,4%	0 0,0%
module_from_spec	6	8 16,3%	0 0,0%
find_spec	29	8 16,3%	0 0,0%
_path_stat	46	7 14,3%	0 0,0%
<built-in method nt.stat>	46	7 14,3%	7 14,3%
_path_is_mode_type	12	2 4,1%	0 0,0%
_path_isfile	10	1 2,0%	0 0,0%
add	200	1 2,0%	1 2,0%
_new_module	5	0 0,0%	0 0,0%
__init__	6	0 0,0%	0 0,0%
acquire	6	0 0,0%	0 0,0%
release	6	0 0,0%	0 0,0%
__init__	6	0 0,0%	0 0,0%
__enter__	6	0 0,0%	0 0,0%

Obrázek 2: Výstup z profilování s 100 vstupními hodnotami

Zde můžeme vidět při profilování s 1000 hodnotami, že ve předu již je více operací, ale zároveň nejvíce času zabírá samostatné otevírání a práce daného skriptu.



Name	Call Count	Time (ms)		Own Time (ms)	
smerodatna_odchylka.py	1	178	100,0%	4	2,2%
<built-in method io.open>	2	113	63,5%	113	63,5%
_find_and_load_unlocked	6	25	14,0%	0	0,0%
_find_and_load	6	25	14,0%	0	0,0%
add	2000	21	11,8%	21	11,8%
_load_unlocked	6	19	10,7%	0	0,0%
math.py	1	16	9,0%	0	0,0%
_find_spec	6	13	7,3%	0	0,0%
find_spec	6	13	7,3%	0	0,0%
_get_spec	6	12	6,7%	0	0,0%
decimal.py	1	12	6,7%	0	0,0%
find_spec	29	11	6,2%	1	0,6%
module_from_spec	6	10	5,6%	0	0,0%
_path_stat	46	9	5,1%	0	0,0%
<built-in method nt.stat>	46	9	5,1%	9	5,1%
subtract	1001	6	3,4%	6	3,4%
power	1000	6	3,4%	6	3,4%
_path_is_mode_type	12	2	1,1%	0	0,0%
_path_join	144	1	0,6%	0	0,0%
_path_isfile	10	1	0,6%	0	0,0%
_path_hooks	2	1	0,6%	0	0,0%
_path_importer_cache	32	1	0,6%	0	0,0%
_new_module	5	0	0,0%	0	0,0%

Obrázek 3: Výstup z profilování s 1000 vstupními hodnotami

3 Zhodnocení

Nejvíce času zabere samotné běhové prostředí pro výpočet, tedy nejvíce sestavení, naplnění daty, načtení knihoven. Je vidět, že matematické operace jsou jednoduché, a zabírají málo času, nicméně při větším množství vstupních dat již znatelný čas zabírají.